Aluno: Kauã Melchioretto
Matéria: Redes de Computadores
Professor: Maykon Chagas

Programação e desenvolvimento de aplicações com TCP

Gaspar 26 de Setembro de 2024

Observações da aplicação

Ao criar os projetos separados e fazer a execução de ambos, foi possível verificar que tanto o lado do client quanto o lado do server precisa ser criado um socket, que através desse socket será fechado as conexões e transferências de dados, tudo precisa ser colocado em um tipo de objeto específico para que chegue corretamente ao lado do servidor e a conexão seja fechada e então tudo funcione como deve funcionar.

Realizei a construção das aplicações em um formato que tanto o client quanto o server ficam em execução para que seja consumido mais de uma vez, fiz loops para que seja mais interessante de fazer as conexões entre client e server. A diferença é que o socket do servidor é de um tipo específico ServerSocket que passa somente a porta que será escutada por parâmetro, já o socket do client será passado o nome do host junto com a porta, como estou na minha local, não sei o nome da minha máquina, bastou apenas passar como “localhost” que ele identifica onde está o servidor.

Achei bem interessante trabalhar com client e server de uma forma tão simples no java, nunca havia feito anteriormente pois não trabalho com java, mas achei extremamente simples a forma a qual com tão poucas linhas de código pode se fazer uma aplicação de cliente e servidor com Java.

OBS: Toda a construção foi feita utilizando a IDE IntelliJ, com um projeto do tipo Maven.

Segue fontes abaixo:
Client:

```java
package org.example;

import java.io.*;
import java.net.Socket;

public class TCPClient {
   public static void main(String[] args) throws Exception {
       String sentence;
       String modifiedSentence;
       boolean run = true;

       BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
       Socket clientSocket = new Socket("localhost", 6789);
       DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
       BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

       System.out.println("-- Para parar a execução do client
digite 'stop' --");

       while(run) {
           System.out.println("Digite uma mensagem: ");
           sentence = inFromUser.readLine();

           if(sentence.equals("stop")) {
               run = false;
               return;
           }

           outToServer.writeBytes(sentence + '\n');
           modifiedSentence = inFromServer.readLine();
           System.out.println("FROM SERVER: " + modifiedSentence);
       }

       clientSocket.close();
   }
}
```

Server:

```java
package org.example;

import jdk.jshell.spi.ExecutionControlProvider;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;

public class TCPServer {
   public static void main(String[] args) throws Exception {
       final int port = 6789;
       ServerSocket welcomeSocket = new ServerSocket(port);

       while (true) {
           System.out.println("Ouvindo na porta: " + port + '\n');
           Socket connectionSocket = welcomeSocket.accept();

           new Thread(() -> {
               try {
                   BufferedReader inFromClient = new
BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
                   DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());

                   String clientSentence;
                   String capitalizedSentence;

                   while ((clientSentence =
inFromClient.readLine()) != null) {
                       capitalizedSentence =
clientSentence.toUpperCase() + '\n';

outToClient.writeBytes(capitalizedSentence);
                   }

                   connectionSocket.close();
               } catch (SocketException e) {
                   System.out.println("Cliente desconectado.");
               } catch (Exception e) {
                   e.printStackTrace();
               } finally {
                   try {
                       connectionSocket.close();
                   } catch (Exception e) {
```

```
                        e.printStackTrace();
                    }
                }
            }).start();
        }
    }
}
```